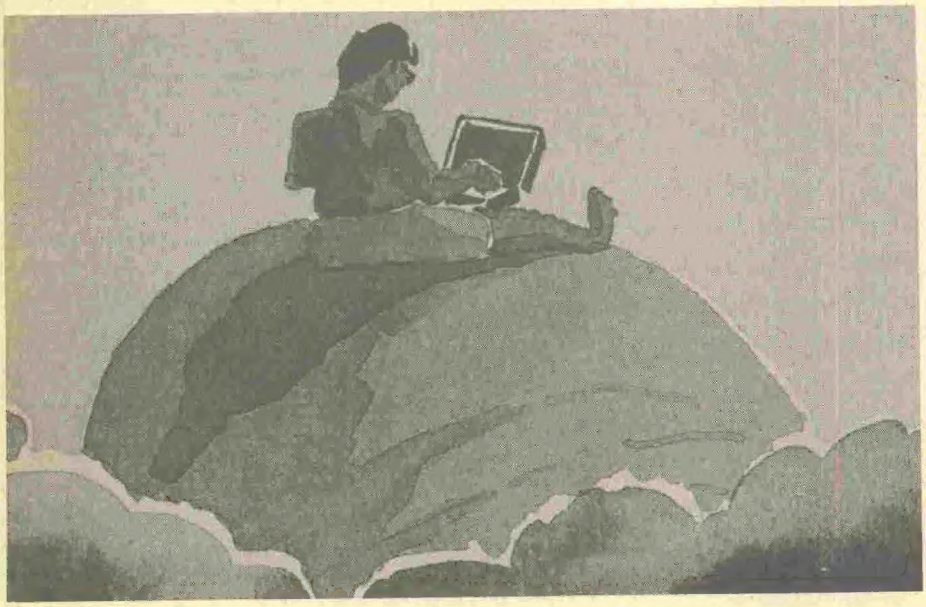
เรื่องนำอ่าน

สรรค์ เสกขุนทด

Three-tier Architecture, Application Partitioning, OOAD และ UML



สถาปัตยกรรมแบบ Three-tier คืออะไร และมีผลกระทบอย่างไร

จนถึงทุกวันนี้ เทคโนโลยี client/server ก็ยังได้รับการกล่าวถึงอยู่เนือง ๆ หลังจากที่ฮิตมาก ๆ อยู่พักหนึ่ง หลังปรากฏการณ์ของเทคโนโลยีนี้ ได้มีคำศัพท์ก่อกำเนิดขึ้นตามมามากมายอันเป็นผลพวงจากการพัฒนาที่ไม่หยุดหย่อน สถาปัตยกรรมแบบ Three-tier คืออะไรและมีผลกระทบอย่างไร Application Partitioning คืออะไรมีประโยชน์อย่างไร ในบทความนี้จะพยายามเสนอก้าวสำคัญในอีกหลาย ๆ ก้าวท่ามกลางกระแสเทคโนโลยีในปัจจุบัน

พูดถึง client/server เสียก่อนเพื่อปูพื้นฐานความเข้าใจเกี่ยวกับ three-tier และ Application Partitioning แนวความคิดของ client/server นั้นเป็นแนวความคิดง่าย ๆ ตรงไปตรงมา (แต่ทำให้ดียาก) นั่นคือแบ่งงานออกเป็นสองฝั่งคือ ฝั่ง server และฝั่ง client ความสัมพันธ์ระหว่าง server และ client ก็คือ client เป็นผู้ขอใช้บริการ และ server เป็นผู้ให้บริการ เมื่อ server ได้รับคำขอนั้นแล้ว ก็จะตอบสนองและอาจส่งข้อมูลกลับ

เครื่อง PC มีส่วนเป็นอย่างมากที่ทำให้แนวความคิด client/server ก่อกำเนิดขึ้นมาได้ เดิมทีนั้นระบบงานคอมพิวเตอร์จะประมวลผลทุกอย่างอยู่ที่ host ซึ่งส่วนใหญ่เป็นเครื่องจำพวกเมนเฟรมหรือมินิคอมพิวเตอร์ เทอร์มินอลมีหน้าที่รับ input และแสดงผลลัพธ์เท่านั้น เมื่อ PC กำเนิดขึ้นมาและมีความเร็วสูงขึ้นเรื่อย ๆ (ที่สำคัญคือถูก) คนก็เริ่มคิดนำ PC มาใช้ให้เกิดประโยชน์สูงสุด หนึ่งในนั้นก็คือหาทางอย่างไรให้ PC ทำงานแทนเครื่องคอมพิวเตอร์โบราณ โดยให้ server ไม่ต้องมีสมรรถภาพอย่างเมนเฟรมและมีหน้าที่เก็บข้อมูลเป็นหลัก ส่วนงานประมวลผลและงานอื่น ๆ ก็ให้ย้ายมาทำบน PC เสีย ดังนั้นเมื่อขนาดคอมพิวเตอร์เล็กลง ราคาถูกลง ตัว PC เองก็นำไปใช้งานอื่นได้อีกด้วย แนวความคิด client/server จึงประสพความสำเร็จอย่างสูง

ในขั้นแรกของการนำแนวความคิด client/server มาใช้โดยเฉพาะงานด้านดาต้าเบสก็เป็นการนำมาใช้แบบตรงไปตรงมา คือ แบ่งงานบางส่วนไว้ที่ server บางส่วนไว้ที่ client แล้วแต่ว่าจะหนักไปด้านใด โดยส่วนใหญ่แล้ว มักจะให้ database server เป็นที่เก็บข้อมูลเฉย ๆ และให้การประมวลผลต่าง การรับคำสั่งและการแสดงผลลัพธ์ให้ย้ายมาทำที่ client แบบที่สองคือ แบ่งงานบางส่วนมาไว้ที่ server เช่น การตรวจสอบข้อมูล การประมวลผล เป็นต้น โดยไว้ในรูปของ data trigger และ stored procedure

แนวความคิดแบบแรกคือ ใช้ PC ทำทุกอย่าง server เก็บข้อมูลเฉย ๆ แนวความคิดแบบนี้สามารถพัฒนาได้เร็วในช่วงต้น ปัญหาที่จะตามมาก็คือ ส่วนที่เป็นกฎเกณฑ์ทางธุรกิจ ที่ฝรั่งเรียกว่า business logic หรือ business rules จะฝังอยู่ที่ client ตัวอย่างกฎเกณฑ์ทางธุรกิจที่ว่าก็ได้แก่ โครงสร้างเงินเดือน เงื่อนไขสวัสดิการต่าง ๆ เงินโบนัส เป็นต้น กฎเกณฑ์

เหล่านี้สามารถเปลี่ยนแปลงได้ตลอดเวลา (ในบางกรณี ขณะเริ่มโครงการใช้อีกอย่าง พอใกล้เสร็จโครงการเปลี่ยนเป็นอีกอย่าง) ดังนั้นเมื่อต้องการเปลี่ยนกฎเกณฑ์เหล่านี้ซึ่งฝังอยู่ที่ client (ตามจำนวน client ที่ใช้) จึงก่อให้เกิดปัญหาอย่างน้อย 2 ประการคือ 1. จะต้องควบคุมกฎเกณฑ์ให้ตรงกันสำหรับทุก ๆ client 2. เมื่อต้องการเปลี่ยน จะต้องตรวจสอบให้ดีว่าจะต้องเปลี่ยนในทุก ๆ ระบบที่ใช้กฎเกณฑ์นั้น พร้อมทั้ง build โปรแกรมใหม่ด้วย

ในทางที่สองคือ พยายามนำกฎเกณฑ์ทางธุรกิจไปไว้ที่ server ให้มากที่สุด ในรูปของ database trigger และ stored procedure วิธีนี้มีข้อดีคือ กฎเกณฑ์อยู่ที่ที่เดียว แก้ไขได้ง่าย ข้อเสียคือมีข้อจำกัดและการพัฒนางานทำได้ยากกว่าวิธีแรก เพราะนักพัฒนาต้องเรียนรู้ data trigger และ stored procedure ของฐานข้อมูลนั้น

ยุคใหม่ของ client/server

ปัญหาดังที่ได้กล่าวมา ถือได้ว่าเป็นปัญหาในยุค two-tier client/server เพื่อที่จะแก้ไขปัญหาดังกล่าว จึงเกิดแนวความคิดที่จะแยกส่วนที่เป็นกฎเกณฑ์ทางธุรกิจออกมาจาก server และ client เป็นอีกหนึ่ง tier มีหน้าที่จัดการเกี่ยวกับกฎเกณฑ์ทางธุรกิจ สถาปัตยกรรมแบบนี้เรียกว่า Three-tier architecture (หรือบางทีก็เรียกว่า n-tied) หลักการก็คือ tier แรกเป็นฐานข้อมูล มีหน้าที่เก็บข้อมูลเป็นหลัก tier ที่สองคือ client มีหน้าที่รับ input และแสดงผล ส่วน tier สุดท้ายเก็บกฎเกณฑ์ทางธุรกิจ ตามรูปที่ 1

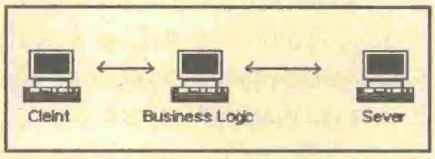
พร้อม ๆ กันนี้ ได้มีการนำแนวความคิดเรื่อง distributed computing และ object-oriented เข้ามาใช้ นั่นคือเป็น distributed object ก็คือว่า business logic tier จะประกอบด้วย object ซึ่งมีหน้าที่รับคำสั่งจาก client และทำการติดต่อกับ server อีกที

Application Partitioning

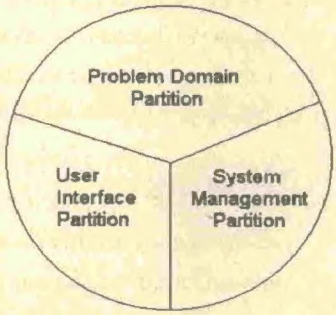
คราวนี้มาดูแนวความคิดอีกอันหนึ่งคือการออกแบบแอพพลิเคชันในลักษณะใกล้เคียงดังกล่าวซึ่งเรียกว่า Application Partitioning คือว่า แอพพลิเคชันหรือระบบหนึ่ง ๆ ประกอบด้วย object ต่าง ๆ มากมาย และแนวความคิดนี้จะแบ่ง object ออกเป็น 3 ประเภท คือ

1. Problem Domain Partition
2. User Interface Partition
3. System Management Partition

object ใด ๆ ก็ตามต้องอยู่ใน partition ใด partition หนึ่ง โดย partition แรกคือ Problem Domain Partition จะประกอบด้วย object ที่ทำงานเกี่ยวกับกฎเกณฑ์ทางธุรกิจเท่านั้น ส่วน partition ที่สอง คือ User Interface Partition ประกอบด้วย object ที่ทำหน้าที่ด้านการรับ input และแสดงผล partition สุดท้ายคือ System Management Partition ประกอบด้วย object ที่จัดการเกี่ยวกับงานที่นอกเหนืองานในทั้งสอง partition แรกที่กล่าวมา อาทิเช่น การจัดการรักษาความปลอดภัย การจัดการความผิดพลาด เป็นต้น (ดูรูปที่ 2)



รูปที่ 1



รูปที่ 2 partition ต่าง ๆ

ข้อเน้นว่า application partitioning เป็นเทคนิคทางด้าน object orinted เท่านั้น เทคนิคนี้ใช้ออกแบบและพัฒนาระบบงาน โดยใช้หลักการของ object orinted คือ ระบบงานประกอบด้วย object ต่าง ๆ มากมาย และเทคนิคนี้จะแบ่ง object ออกเป็น 3 ส่วนดังกล่าวข้างต้น

Three-tier และ Application Partitioning

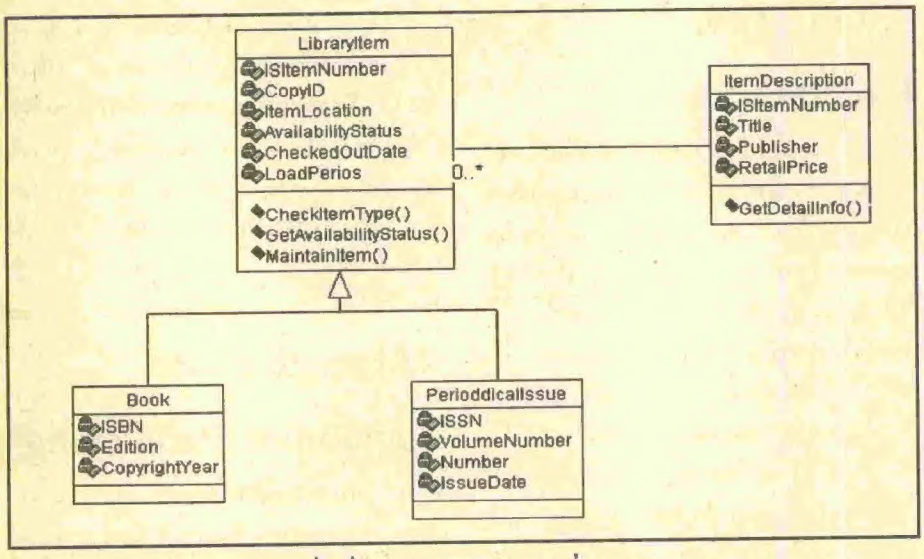
แนวความคิดทั้งสองมีความคล้ายคลึงกันอยู่มาก three-tier ต้องการให้แยกกฎเกณฑ์ทางธุรกิจออกจาก client และ server เช่นเดียวกัน Application Partitioning ก็แยก Problem Domain ออกมาต่างหาก client ทำหน้าที่จัดการ input/output ใน three-tier เหมือนกับ User Interface Partition

เจาะลึก Application Partitioning

หากว่าต้องการนำเทคนิค Application Partitioning ไปใช้จะมีขั้นตอนอย่างไร แรกสุดก็คือต้องหาและดี requirement ให้แตก ผลลัพธ์ที่ได้คือ class diagram ของ Problem Domain Partition ในรูปที่ 3 แสดงตัวอย่าง class diagram ระบบห้องสมุด

เนื่องจากว่า Application Partitioning เป็นเทคนิคด้าน object-oriented ดังนั้นสิ่งที่ขาดไม่ได้ในการอธิบายความสัมพันธ์ระหว่าง object ต่าง ๆ ก็คือ class diagram นั่นเอง

ในการวิเคราะห์ระบบงาน สิ่งที่ได้จากการวิเคราะห์คือความรู้ในเนื้องาน ความรู้นี้เองที่นักวิเคราะห์จะต้องนำไปเขียน class diagram เพื่อแสดงถึง object พื้นฐานกับความสัมพันธ์ระหว่าง object เหล่านั้น รวมทั้ง attribute และ operation หลัก ๆ หลังจากนั้นในช่วงออกแบบจึงใส่รายละเอียดมากขึ้น เช่น data type ของ attribute พารามิเตอร์ของ operation เป็นต้น



รูปที่ 8 ตัวอย่าง class diagram สี่เหลี่ยมแต่ละอันคือคลาสหนึ่ง ๆ

Application Partition แบ่ง object ทั้งหมดออกเป็น 3 partition เพราะฉะนั้นในแต่ละ partition จะต้องมี class diagram ของตัวเอง สำหรับ User Interface Partition และ System Management Partition เราอาจสร้างเองได้หรือหาซื้อตามท้องตลาด แต่เราต้องสร้าง class diagram เองสำหรับ Problem Domain Partition เพราะมีความเกี่ยวพันกับเนื้องานโดยตรง

เมื่อได้ class diagram ของ Problem Domain Partition แล้ว ขั้นตอนต่อไปคือแปลงคลาสที่ได้ไปเป็น relational table วิธีการคร่าว ๆ คือ แปลงคลาสไปเป็นเทเบิ้ล แปลง attribute เป็นฟิลด์ และเรคอร์ดใด ๆ ในเทเบิ้ลก็คือ object แต่ละชิ้นของคลาสนั้น

สาเหตุที่ต้องแปลง class diagram ก็เพราะว่า ในปัจจุบันยังมีการใช้ relational database ค่อนข้างมาก ส่วน object oriented database นั้นก็มีออกมาบ้าง แต่ยังไม่เป็นที่ยอมรับในวงกว้าง

OOAD และ UML

OOAD ย่อมาจาก Object Oriented Analysis and Design ซึ่งก็คือ กรรมวิธีวิเคราะห์และออกแบบงานโดยใช้หลักการ object oriented มีกรรมวิธีในการวิเคราะห์และออกแบบระบบมากมาย แต่ OOAD เป็นกรรมวิธีล่าสุด และคาดว่า ในอนาคตจะมาแทนกรรมวิธีเดิมที่ใช้กันอยู่

OOAD ได้รับการวิจัยและพัฒนามานานพอสมควร ผู้ที่เป็นหัวหอกสำคัญได้แก่ Rumbaugh, Jacobson และ Booch ทั้งสามคนได้คิดค้นกรรมวิธี OOAD ขึ้นมาตามแนวความคิดของตน มีทั้งส่วนเหมือนและส่วนต่าง มีทั้งจุดแข็งและจุดอ่อน อย่างไรก็ตาม เป็นที่น่ายินดีว่า ทั้งสามคนนี้ได้มารวมพลังสร้างมาตรฐานของ OOAD ขึ้นมาที่มีชื่อว่า UML (Unified Modeling Language) และขณะนี้ UML กำลังอยู่ในช่วงพิจารณาจากคณะกรรมการของ OMG คาดว่า UML คงจะผ่านและกลายเป็นมาตราฐานต่อไป

UML และ Application Partititoning

UML ถูกออกแบบมาเพื่อสร้างระบบใด ๆ ส่วน Application Partitioning เป็นเทคนิคทาง object oriented ที่ใช้สร้างระบบงานที่เกี่ยวกับฐานข้อมูล เหตุดังนั้น UML และ Application Partitioning จึงไปกันได้ด้วยดี พูดอีกนัยหนึ่งก็คือว่า Application Partitioning นำ UML มาใช้ในด้านที่จำเพาะเจาะจงลงไป และหากว่า UML ประสพความสำเร็จอย่างสูง Application Partitioning ก็จะได้รับการยอมรับและมีการพัฒนาเป็นลำดับต่อไป

ผลกระทบ

ประโยชน์ของ Applitcation Partitioning ก็คือประโยชน์ที่ได้รับจากแนวความคิด object oriented นั่นเอง ที่สำคัญได้แก่ ความสามารถในการนำกลับมาใช้ใหม่ได้ (reuseablity) ความสามารถในการต่อเติม (extendility) และอื่น ๆ ประโยชน์ของ object oriented

ในขณะนี้มีการนำแนวความคิด object-oriented ไปใช้อย่างแพร่หลาย เช่น สถาปัตยกรรม ActiveX/DCOM ของไมโครซอฟท์ หรือสถาปัตยกรรม COBRA ของ OMG หรือ OOAD ก็ตาม ในอนาคตเราคงได้เห็นการเติบโตของหลายด้านที่ว่า โดยเฉพาะ OOAD ถึงวันนั้น OOAD อาจกลายเป็นมาตรฐานในการวิเคราะห์และออกแบบระบบงาน ถึงวันนั้นจริงนักคอมพิวเตอร์ไทยทั้งหลายคงต้องปรับตัวกันอย่างมโหฬาร มิฉะนั้นแล้วเรา (หมายถึงคนในวงการคอมพิวเตอร์ไทย) อาจจะต้องล้าหลังอีกไม่รู้เท่าไหร่ปี เหมือนอย่างที่นายทุนไทยพ่ายแพ้สงครามเศรษฐกิจในโลกยุคไร้พรมแดน